

In the Specification:

The paragraph beginning page 8, line 2 is amended as follows:

FIG. 12 depicts a hardware and software environment in which the Portlet Template of the present invention operates. In FIG. 112 12, an object oriented computing environment 111 operates on one or more computer platforms 112. Object oriented computing environment 111 includes an object manager. Computer platform 112 may include computer hardware units 113 such as a central processing unit (CPU) 114, a main memory 115, and an input/output (I/O) interface 116, and may further include peripheral components such as a display terminal 121, an input device 122 such as a keyboard or a mouse, nonvolatile data storage devices 123 such as magnetic or optical disks, printers 124 and other peripheral devices. Computer platform 112 may also include microinstruction codes 126 and an operating system 128. Operating system 128 may exist within the object oriented computing environment 111 and may be written in the Java object oriented programming language.

The paragraph beginning page 11, line 15 is amended as follows:

A state design pattern is an object oriented design embedded in the Portlet Template. Using the state design pattern, the Portlet Template provides the following components to quickly and reliably create a portlet with the added benefit of allowing the developer to more easily modify code as a result of changing requirements or design: Template Java classes for states (e.g., State classes for View, Edit, Configure, and Help modes); Template Java classes for actions (e.g., Action classes for View, Edit, Configure, and Help modes); and a base portlet and base controller library (JAR) that provides a foundation and project specific standard

functionality (e.g. error logging, significant event handling, etc.).

The paragraph beginning page 13, line 18 is amended as follows:

The block diagram of FIG. 3, which describes portlet code development with the Portlet Template, includes blocks 21-26 21-25. In block 21, the portlet developer writes code in the controller to set an initial state as the current state. In block 22, the portlet developer writes the State classes containing **perform view** codes to display the J.P. page associated with the each state. In block 23, the portlet developer writes action listener code in the J.P. code to generate a page with action links, and for each action link the developer must set the action to be a specific Action class name that performs the intended action for that link. In blocks 24A-24C, the portlet developer writes the Action classes containing **action performed** codes to execute the business methods that perform the intended actions of the user. Note that blocks 24A-24C denote multiple business methods inasmuch as the portal may include multiple links, wherein each such link has an associated intended action (i.e., business method) to be performed. For each action performed, in blocks 25A-25C the portlet developer writes a **set state** method in the Action class that determines one destination state of the State class, wherein the destination state (called the “next state”) includes a **perform view** display method to be executed for displaying the next page following performance of the intended action in step 24C. Following performance of the steps of blocks 24-25, the portlet developer would repeat the steps of blocks 23-25 for the next state and action.

The paragraph beginning page 15, line 20 is amended as follows:

The program code includes a portlet code module (BasePortlet and TemplatePortlet) and a controller code module (BaseController and TemplateControllerForHtml) in the program code. The BasePortlet class of the portlet code module is adapted to execute the **action performed** method and the **set state method**, and the BaseController class of the controller code module is adapted to execute the **perform view** method.